

ASSIGNMENT-3.3

2303A51295

BATCH-10

TASK-1:

PROMPT:

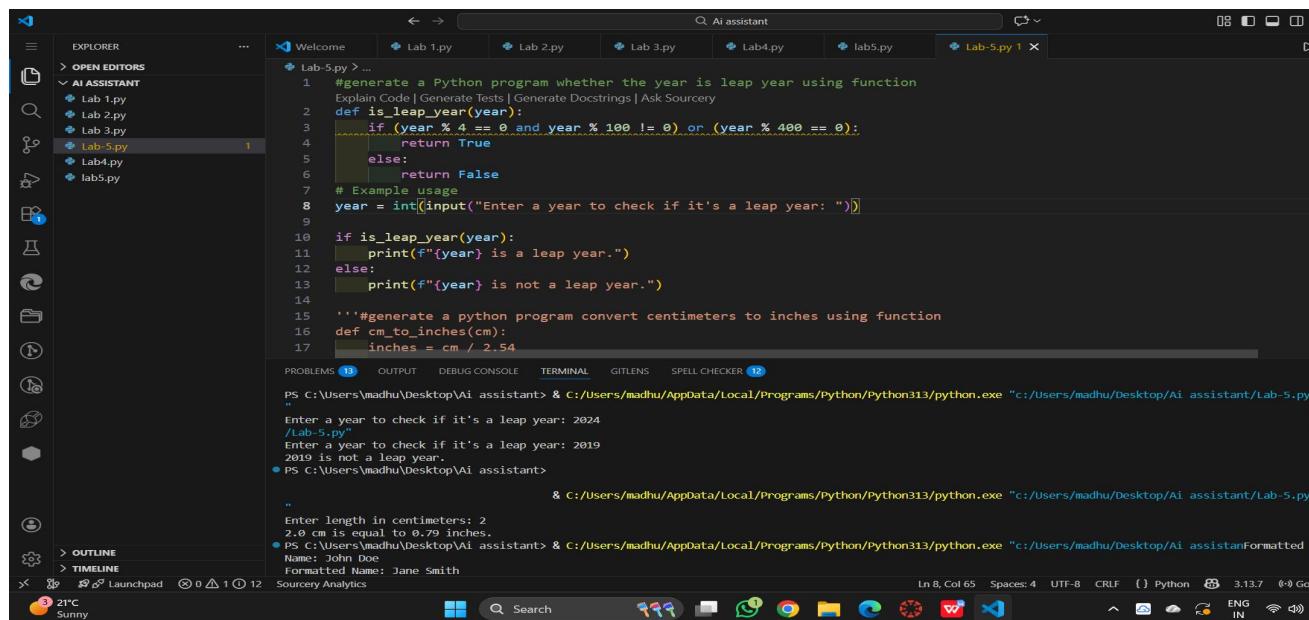
generate a Python program whether the year is leap year using function

CODE:

```
def is_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year %
400 == 0):
        return True
    else:
        return False
# Example usage
year = int(input("Enter a year to check if it's a leap
year: "))

if is_leap_year(year):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.)
```

OUTPUT:



The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER:** Shows files Lab 1.py, Lab 2.py, Lab 3.py, Lab 4.py, Lab5.py, and Lab-5.py (the active file).
- AI ASSISTANT:** A sidebar with various AI-related icons.
- CODE EDITOR:** Displays the Python code for determining leap years.
- TERMINAL:** Shows the command-line output of running the script and interacting with the user to enter a year.
- STATUS BAR:** Shows the current file path (c:/Users/madhu/Desktop/Ai assistant/Lab-5.py), line count (Ln 8, Col 65), and other system information.

EXPLANATION:

- This program determines whether a given year is a leap year using a function.

- The function applies standard leap year rules and returns True or False.
- The user inputs a year, which is checked by the function
- The result is printed as either a leap year or not.

TASK-2

PROMPT:

generate a python program convert centimeters to inches using function

CODE:

```
def cm_to_inches(cm):
    inches = cm / 2.54
    return inches
# Example usage
cm_value = float(input("Enter length in centimeters: "))
inches_value = cm_to_inches(cm_value)
print(f"{cm_value} cm is equal to {inches_value:.2f} inches.")
```

OUTPUT:

The screenshot shows a Visual Studio Code (VS Code) interface. The left sidebar has icons for Explorer, Open Editors, AI Assistant, and others. The Explorer panel shows files like Lab 1.py, Lab 2.py, Lab 3.py, Lab 5.py (selected), Lab4.py, and lab5.py. The main editor area displays Python code for checking if a year is a leap year and for converting centimeters to inches. The terminal at the bottom shows command-line interactions for file operations like 'mv' and 'cp'. Status bar at the bottom right shows file count, spaces, and encoding.

```
if is_leap_year(year):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")

#generate a python program convert centimeters to inches using function
def cm_to_inches(cm):
    inches = cm / 2.54
    return inches

# Example usage
cm_value = float(input("Enter length in centimeters: "))
inches_value = cm_to_inches(cm_value)
print(f'{cm_value} cm is equal to {inches_value:.2f} inches.')

'''#generate a python program to name formatting using function with multiple examples
def format_name(full_name):
    pass

""

Enter a year to check if it's a leap year: 2024
/Lab-5.py"
Enter a year to check if it's a leap year: 2019
2019 is not a leap year.
● PS C:\Users\madhu\Desktop\AI assistant>

& C:/Users/madhu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/madhu/Desktop

"
Enter length in centimeters: 2
2.0 cm is equal to 0.79 inches.
● PS C:/Users/madhu/Desktop/AI assistant> & C:/Users/madhu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/madhu/Desktop
Name: John Doe
Formatted Name: Jane Smith
Formatted Name: Alice O'Connor

Sourcey Analytics
```

EXPLANATION:

- This program converts a length from centimeters to inches using the correct mathematical formula.
 - A function performs the conversion by dividing the value by 2.54.
 - The user enters a value in centimeters, which is passed to the function.
 - The converted result is displayed in inches.

TASK-3:

PROMPT:

#generate a python program to name formatting using function with multiple examples

CODE:

```
def format_name(full_name):
```

```
parts = full_name.split()
```

```
if len(parts) >= 2:
```

```
first_name = parts[0]
```

```
last_name = parts[-1]
```

```
return f'{last name}, {first name}'
```

else:

```
return full_name # Return as is if not enough parts
```

```
# Example usage
```

test names = [

"John Smith".

"Anita Rao".

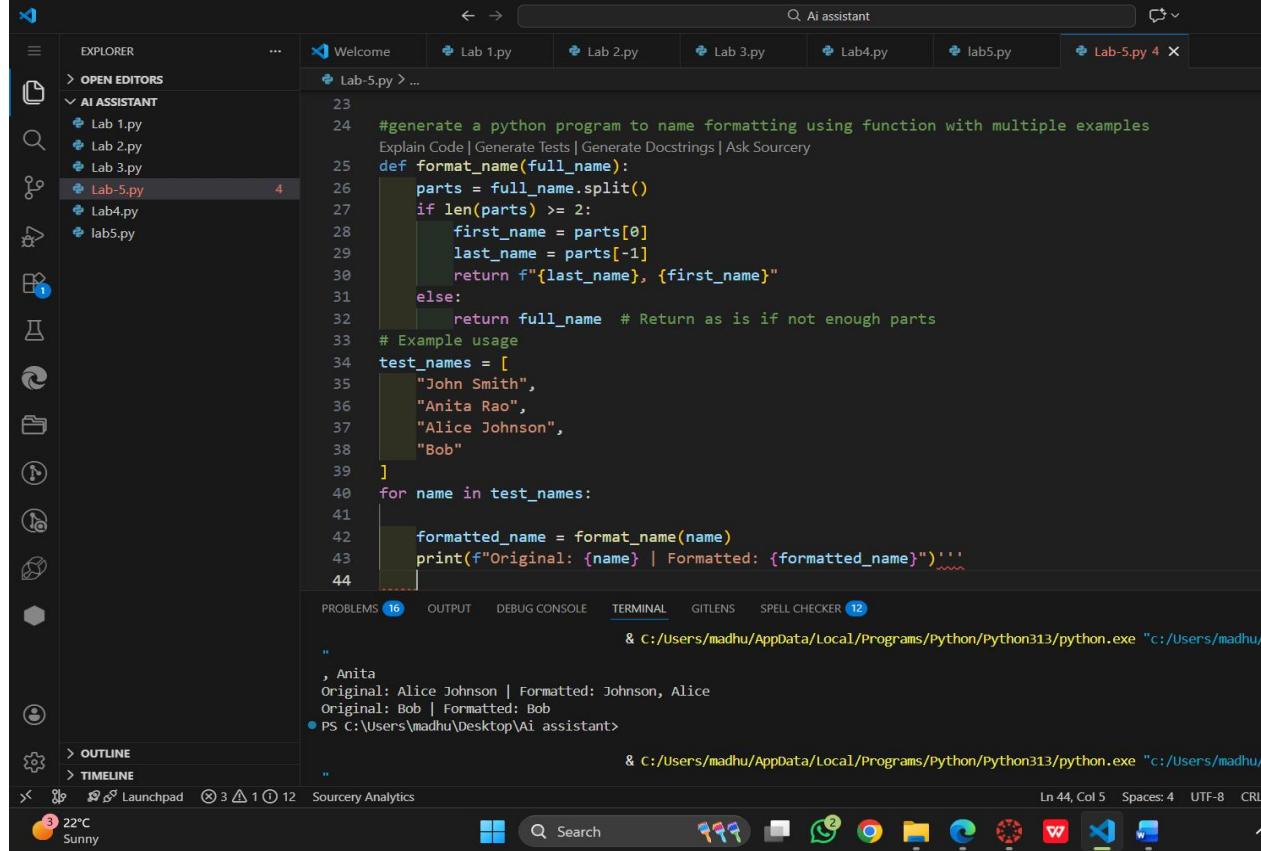
"Alice Johnson".

"Bob"

]

for name in test_names:

```
    formatted_name = format_name(name)
    print(f"Original: {name} | Formatted: {formatted_name}")
```



The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows files Lab 1.py, Lab 2.py, Lab 3.py, Lab 4.py, lab5.py, and Lab-5.py (the active file).
- Code Editor:** Displays the Python code for a function `format_name` and a loop that prints formatted names.
- Terminal:** Shows the command `& C:/Users/madhu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/madhu/"` and the output of the program running on Windows 10.
- Bottom Status Bar:** Shows the date (22°C, Sunny), system icons, and file paths.

OUTPUT:

EXPLANATION:

- This program formats a full name entered as first name and last name.
- The input is split and validated to ensure the correct format.
- Each part of the name is cleaned and capitalized.
- The formatted full name is then displayed.

TASK-4:

PROMPT:

generate a python code using function that counts vowels in a string

CODE:

```
def count_vowels(input_string):
    vowels = "aeiouAEIOU"
    count = sum(1 for char in input_string if char in vowels)
```

```
        return count
# Example usage
test_string = "Hello World"
vowel_count = count_vowels(test_string)
print(f"The number of vowels in '{test_string}' is: {vowel_count}")
```

OUTPUT:

The screenshot shows the Visual Studio Code interface with the AI Assistant extension active. In the Explorer sidebar, files like Lab 1.py through Lab 5.py are listed. The main editor tab is titled 'Lab-5.py' and contains the following Python code:

```

38     "Bob"
39 ]
40 for name in test_names:
41
42     formatted_name = format_name(name)
43     print(f"Original: {name} | Formatted: {formatted_name}")
44
45
46
47
48 #generate a python code using function that counts vowels in a string
49
50 def count_vowels(input_string):
51     vowels = "aeiouAEIOU"
52     count = sum(1 for char in input_string if char in vowels)
53     return count
54
55 # Example usage
56 test_string = "Hello World"
57 vowel_count = count_vowels(test_string)
58 print(f"The number of vowels in '{test_string}' is: {vowel_count}")
59
#generate a python program with function that reads the file and counts no.of lines in the file

```

The status bar at the bottom indicates the code is being generated by the AI assistant.

EXPLANATION:

- The function counts vowels in a given string using a direct logic approach.
- Zero-shot prompting applies the logic without examples.
- Few-shot prompting helps by showing patterns before execution.
- The function returns the total number of vowels in the input string.

TASK-5:

PROMPT:

generate a python program with function that reads the file and counts no.of lines in the file and return the line count

CODE:

```

def count_lines_in_file(file_path):
    try:
        with open(file_path, 'r') as file:
            lines = file.readlines()
            return len(lines)
    except FileNotFoundError:
        print(f"Error: The file '{file_path}' was not found.")
        return None
# Example usage
file_path = input("Enter the file path to count lines: ")
line_count = count_lines_in_file(file_path)

```

if line_count is not None:

```
    print(f"The number of lines in the file '{file_path}' is: {line_count}")
```

OUTPUT:

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER:** Shows files in the AI ASSISTANT folder: Lab 1.py, Lab 2.py, Lab 3.py, Lab 4.py, and Lab 5.py. Lab 5.py is the active editor.
- EDITOR:** Displays the code for Lab 5.py. The code defines a function `count_lines_in_file` that reads a file and counts the number of lines. It includes a docstring and a usage example.
- TERMINAL:** Shows command-line output for running the script. It shows an error for a non-existent file and then successfully runs the script on a valid file, printing the line count.
- STATUS BAR:** Shows system information like weather (22°C, Sunny), battery level, and system date (28-01-2026).

EXPLANATION:

- This program reads a text file and counts the number of lines.
- A function opens the file safely and calculates the line count.
- Error handling is used if the file does not exist.
- The final line count is returned and displayed.